Humboldt Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät

Institut für Informatik

Bridging the Reality-Gap: 6-DoF Pose Estimation of Multiple Cars by a Deep CNN Trained on Synthetic Data

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

eingereicht von: Benjamin Lorenz
geboren am: 06.01.1988
geboren in: Königs Wusterhausen
Gutachter Prof. Dr. Ralf Reulke
Dr. Andreas Leich

eingereicht am: _____

verteidigt am: _____

Abstract

Precise localization in the form of 6-DoF pose estimation is of great value in traffic scenario research. Convolutional neural networks yield robust, fast, and precise results but need huge amounts of annotated data. Obtaining such data in the real world is tedious and error-prone. In contrast, simulation allows automation of the data generation process by creating unlimited amounts of fully and correctly annotated synthetic data. But due to the lack of realism, networks trained on this data usually do not perform well on real data. Recent works bridged this so-called reality gap with synthetic data of high diversity, utilizing domain randomization and photorealistic synthetic data. The Deep Object Pose Estimation (DOPE) shows convincing results in detecting multiple different household objects for the task of robotic grasping.

In the thesis I demonstrate that DOPE can be used for the domain of traffic monitoring, in which multiple objects of the same class, namely cars in traffic, are involved. For this purpose I create the synthetic 6-DoF pose estimation dataset MultiCarPose containing highly diverse scenes and utilize it for training DOPE. This allows me to show, that the domain change from robotic grasping to traffic monitoring is possible, that the trained model is able to perform the transition from synthetic to real data and that the pose of multiple objects of the same class in one image can be estimated.

Contents

List of Figures

List of Tables

List of Abbreviations

6-DoF ..... Six Degrees of Freedom
CNN ..... Convolutional Neural Network
DOPE ..... Deep Object Pose Estimation
FN ..... false negative
FP ..... false positive
MCP ..... MultiCarPose
NPC ..... Non-Player Character
PnP ..... Perspective-n-Point
RGB ..... RGB color model
TP ..... true positive
YCB ..... Yale-CMU-Berkeley Benchmarks - Object and Model Set

1 Introduction

Precise localization is of great value in traffic scenario research. Extracting as much information as possible from the environment with cheap and simple sensors is the key in these domains. Especially six degrees of freedom (6-DoF[1] pose estimation from monocular RGB images, and still a challenging task and active research question in computer vision.

Recent 6-DoF pose estimation approaches[34, 16, 41, 39] show promising results utilizing deep convolutional neural networks. The performance of these networks is based on large amounts of annotated data though. Due to the many variables present in a 6-DoF pose, the process of annotating such data in the real world, namely fitting a projected cuboid around an object, in image space is tedious and error-prone. Therefore, existing traffic related datasets either geographic modelling[20] or utilize expensive and complex multi sensor systems[11, 33, 10, 4, 1, 17, 21, 24] to obtain ground truth pose information. But these datasets have two shortcomings. NYC3D-Cars[20] is with 5K images rather small and mostly covers pedestrian-level perspectives, while the latter were created for the task of autonomous driving and are therefore limited to the car-level perspective. Besides these restrictions, the training data is highly correlated with test data (same camera, same object instances, similar lighting conditions) in these datasets[39]. This leads to bad generalization capabilities of networks trained with this data [39].

In contrast to real data, simulation allows automation of the data generation process. Synthetic data can be generated automatically in large quantities and without any errors. Furthermore, they are not subject to any restrictions regarding privacy related data. Previous works on synthetic data overcame complex challenges like semantic segmentation[27, 18], viewpoint estimation[31] and also pose estimation[39]. But synthetic data comes with its own problems. The reality gap, resulting from the lack of realism, leads to the fact that networks trained on synthetic data usually do not perform well on real world data without additional fine-tuning[39, p. 1]. In this thesis, I want to utilize the Deep Object Pose Estimator (DOPE)[39] to estimate the pose of multiple cars in traffic from a traffic monitoring perspective. DOPE[39] is based on a deep convolutional neural network and was originally developed for robotic grasping. It is capable of estimating the pose of multiple different household objects from monocular RGB images without the use of additional information, like point clouds or depth information.

Not only do I transfer DOPE[39] to another domain but also modify the use case. Instead of multiple different objects, I want to estimate the pose of multiple objects of the same type, namely cars. Pose estimators based on deep convolutional neural networks have the advantage over geometric approaches, that they do not require

[1] Refers to the freedom of movement of a rigid body in 3D space.

assumptions about the surroundings and can adapt more flexible to new environments.

Because the pose estimation data is scarce in the traffic monitoring domain, I create the MultiCarPose (MCP) dataset to make this transfer possible. MCP is a synthetic pose estimation dataset which I use for the training of DOPE[39]. The test and validation set of MCP consists of synthetic data that I created with the help of CARLA[8] and NDDS[35]. The former is a traffic simulation with which I create photorealistic images that are similar to reality in many aspects. The latter allows me to create domain randomized[36] images that try to create as much diversity as possible with random backgrounds, random textures, random positions etc.. In order to measure and evaluate the performance of the model generated by the training process, I annotate projected cuboids for the test set manually. As already noted, I decided to do this because there is a lack of pose estimation data for this domain. For the assignment of an estimated projected cuboid to a ground truth, I use the Kuhn-Munkres algorithm together with the ADD-2D which is part of the contribution of this work, based on the average distance metric (ADD)[3] to measure the distance of projected cuboids in image space. I have to use a metric to estimate poses in image space, because I do not have any real ground truth poses. Apart from small changes for the evaluation, I use DOPE[39] as an off-the-shelve application, since it is, with its near realtime performance, currently the most promising single shot pose estimator. In order to bridge the reality gap, I decided to use a combination of domain randomized and photorealistic data in the MCP dataset. On the one hand, Tremblay et al.[39] were able to achieve very good results with such an approach. On the other hand, I assume that the greater the variability of the synthetic data is, from which a neural network learns, the easier the adaptation to reality will succeed.

In the following I explain central terms of this thesis and describe the scientific status quo and related work in the context of pose estimation networks and the utilization of synthetic data for the training of neural networks. I then disuss my research question in more detail and explain my approach and the methods chosen to answer it. I will then go into the principles of how DOPE works and describe the structure, composition and process of creating the MCP dataset. Subsequently, I go into individual details of the training and explain the structure of my measurements, assumptions and observations in the evaluation, and finally discuss the results.

2 Basics

In this chapter I would like to lay the foundation to approach the thesis. There are some terms which can have different meanings depending on the context. In order to avoid ambiguity, I describe central terms in this chapter. Subsequently, I present the current state of research which this thesis is based on. In particular, I present the current state of pose estimation networks and synthetic data in connection with these

networks.

2.1 Terms

In this section I would like to introduce key concepts and clearly define them.

6-DoF Pose Estimation The concept of 6-DoF pose estimation describes the task to identify specific objects in an image and to determine each object's position and orientation relative to the camera coordinate system. 6-DoF refers to the freedom of movement of a rigid body in 3D space. Specifically, the body is free to change position as forward/backward, up/down, left/right translation in three perpendicular axes, combined with changes in orientation through rotation around three perpendicular axes, often named yaw , pitch, and roll. In the course of the work I will refer to 6-DoF pose estimation as pose estimation.

Perspective-n-Point Perspective-n-Point (PnP) describes the problem of estimating the pose of a camera by a set of 3D points in the world and their corresponding 2D projections in the image. DOPE[39] utilizes PnP, more precisely P8P, to determine the pose of an object from the eight points of the projected cuboid in the image.

Traffic Monitoring Scenario By the term traffic monitoring scenario I refer to a scenario in which events on and beside the road are captured from a static position outside of vehicles. For example, at infrastructure such as traffic lights, bridges, posts or buildings. This is done in distinction to the scenario of autonomous driving, in which the events are captured from a moving vehicle. The difference is relevant as cars appear in different ways in the images in those scenarios. So that, a neural network that has only been trained on images from the car perspective, has difficulties to recognize a car from above.

Definition Car The concept *car* should be generally known. However, there are many different types of vehicles and some of them require special consideration. Following the Cityscapes dataset[6] definition, a car is a vehicle with four wheels that is designed to transport people. Accordingly, jeeps, SUVs and vans are also cars as long as they have a continuous body shape and no trailers[6]. In distinction, a vehicle is considered a truck if the back part is physically separated from the driving compartment and a bus is for more than nine persons. In this thesis only the pose of cars is estimated.

Cuboid or 3D Bounding Box In connection with Pose Estimation the term 3D bounding box is often used. The term is derived from the two dimensional bounding box in object detection. The problem is that in the course of the paper the same terminology is used for different definitions. So the term bounding box is used as a

synonym for 3D bounding box. This can lead to confusion especially if the projected 3D bounding box is involved in the image space, where possibly a 2D bounding box would make sense in some cases, too. That is why I use the term projected cuboid when I am talking about the eight points around an object in the image space and thus each point has two coordinates, namely (x, y). If the eight points are about an object in the world or camera coordinate system I use the term cuboid. Then each of the eight points has three coordinates (x, y, z).

2.2 Related Work

The task of pose estimation of multiple cars with a CNN trained on synthetic data is tangent to two questions. What is the state of the art with regard to pose estimation networks and with regard to the training of neural networks with synthetic data for the pose estimation tasks?

6-DoF Pose Estimation Networks

In this section I want to focus on the research on pose estimators based on CNNs which use single RGB images as input without any depth, stereo vision or point cloud information. These networks get a single RGB image as input and return the rotation and translation of detected objects in the world or camera space. The first generation of deep learning approaches estimating 6-DoF poses in this way[25, 16, 34, 41] surfaced in 2017 and were able to show improved performance handling difficulties such as occlusion compared to previous approaches.

In[25] Rad and Lepetit proposed a cascade of multiple CNNs for the object pose estimation task, called BB8. It first localizes objects with a segmentation network and predicts the 2D projections of the 3D bounding boxes of the objects in the second step. From the correspondences between the projected 2D coordinates and the 3D bounding box, the 6-DoF pose is estimated, with the help of a PnP algorithm, followed by an optional refinement step, in which one CNN is trained per class[25]. Therefore BB8 is not an end-to-end approach and the complex architecture slows down the inference[7].

In SSD-6D[16] Kehl et al. extend the SSD[19] single-shot object detector framework to predict 6-DoF poses. They decomposed the 3D rotation space into discrete viewpoints and in-plane rotations and treat the rotation estimation as a classification problem[7]. As a first step the object's 2D bounding boxes and a coarse estimate of the orientation is estimated, followed by a depth prediction step utilizing the size of the 2D bounding box. A further refinement step is then necessary to improve the pose's accuracy.

Xiang et al.[41] propose a method called PoseCNN, which decouples the estimation of 3D rotation and 3D translation. The network estimates the 3D translation of an object by localizing its center in the image and predicting its distance from the camera and

regressing the 3D rotation to a quaternion representation[41]. Moreover they designed a new loss function, which allows PoseCNN to handle occlusion and symmetric objects in cluttered scenes. By relying on a semantic segmentation approach to localize objects, PoseCNN might have problems to deal with multiple instances of an object in an image[7]. Moreover the network needs depth information to further refine the object's pose. Also BB8 and SSD-6D [25, 16] require further post-refinement steps to increase accuracy[7] which slows down inference and doesn't allow real-time performance of multiple objects.

Do et al[7] used a Mask R-CNN[12] as the foundation for their Deep-6DPose network. They added an additional branch to the backbone of Mask R-CNN[12] which takes regions of interest as inputs to regress the 6D object poses. As a result Deep-6DPose is an end-to-end architecture which is not only detecting, segmenting but also directly recovering the 6D poses of object instances from a single RGB image without post-refinement[7]. But with an inference performance of 10 fps Deep-6DPose is not capable of real-time estimation.

Tekin et al.[34] proposed a single-shot 6D pose prediction architecture that extends the single shot 2D object detection paradigm to 6D pose estimation and detection with real-time performance[43, 34]. Similar to SSD-6D[16], Tekin et al.[34] upgraded the YOLO single-shot object detector[26] to estimate 2D projections of 3D bounding boxes around objects. They then apply a PnP algorithm to compute rotation and translation to obtain 6-DoF. As there is no post refinement step necessary, there is no overhead for pose estimation of multiple objects although the real-time capability is only shown for single objects in[34]. As this method is based on YOLO[26], which has problems detecting close objects from the same class, it can be assumed that Tekin et al.[34] will have problems regarding closeness in pose estimation, too.

Song et al.[30] just published their method called HybridPose, which is yielding outstanding results on single object pose estimation on the LineMod dataset[13] and even for multiple objects on Occlusion dataset[3] while being real-time capable with 30fps. HybridPose[30] utilizes a combination of keypoints, edge vectors, and symmetry correspondences. Unfortunately it cannot handle multiple instances of the same class in one image.

PVNet[22] and DPOD[42] can handle multiple instances in one image but need accurate 3D models of the objects for pose estimation. Due to the many different types of cars it is not feasible to provide a 3D model for each possible type of car.

Tremblay et al.[39] contributed their DOPE network which uses the first ten layers of VGG-19[29] as a feature extractor followed by a two-step solution to estimate the 6-DoF pose. The vertices of the projection of the 3D bounding box are estimated with the help of belief maps and vector fields. These vertices are then used to predict the final pose using PnP, assuming known camera intrinsics and object dimensions. The network is able to perform pose estimation of multiple different objects in near

real-time, which is why I chose it for this thesis.

Utilizing Synthetic Data for CNNs

Utilizing synthetic data to train deep CNNs overcame complex computer vision tasks like object detection[36, 32, 2, 14, 23, 37, 40], semantic segmentation[18, 27], viewpoint estimation[31], and even pose estimation[15, 39].

For the task of viewpoint estimation, which is related to the pose estimation problem, Su et al.[31] proposed an image synthesis pipeline. For an input RGB image and a bounding box from an object detector, their goal is to estimate camera rotation parameters, namely the azimuth, elevation and in-plane rotation angles. Su et al.[31] utilized images of 3D models, viewed from every direction on random background, with randomly changed sampling parameter and applied truncation patterns. Instead of pursuing realistic effects, they focused on generating a diversity of images. Their network was then trained on a combination of 12K images from VOC12[9] training set and 2.4M synthetic images and could significantly outperform existing methods[31].

With the goal of robotic grasping Josifovski et al.[15] and Tremblay et al.[39] proposed two methods of 6-DoF pose estimation trained solely on synthetic data. Josifovski et al.[15] used the bounding boxes extracted with an object detector as input for a viewpoint estimation model, which calculates the pose. Kehl et al.[16] treated the pose estimation task as a classification problem and created two separate datasets. One for the object detection task, created with 3D models in Blender, randomizing their position on random background. Followed by utilizing separate datasets for each object in all possible angles, annotated with azimuth, elevation and in-plane rotation. They found that the detector can learn from synthetic data and that the detector can generalize from unrealistic rendered data as good as from realistically rendered data. Tremblay et al.[39] aimed to infer the 6-DoF pose of objects in clutter, from a single RGB image with a deep CNN trained on synthetic data. For their dataset they used a mixture of photorealistic synthetic data and domain randomized data[36]. In the latter the lighting and textures are rendered in a non-realistic ways, but by randomizing these parameters also in very divers ways. Both datasets were created by placing YCB[2] object models[5] in different visual environments. For the domain randomized dataset the YBC objects were put on random backgrounds and arranged next to or behind distractor objects in random lighting conditions with noise and random object poses. The photorealistic dataset on the other hand was generated by placing foreground objects in 3D background scenes with physical constraints. The background was chosen from Unreal4 Engine virtual environments with high fidelity modeling and quality. The YCB[5] objects then fall under the influence of gravity and collide with each other and with the surface. The authors published this dataset under the name

[2]YCB (Yale-CMU-Berkeley) is an object and model set for benchmarking robotic manipulation.

Falling Things[38]. To train their 6-DoF estimator Tremblay et al.[39] used 60k photorealistic images mixed with 60k domain randomized images per class. They found that the combination of photorealistic and domain randomized images yields sufficient diversity and complexity to train a deep CNN that is then able to operate on real data without fine-tuning. The entirely on synthetic data trained pose estimator was able to infer the pose of known objects in clutter in near real time, which is a performance comparable with state-of-the-art networks trained on real data.

3 Research

In the following section I formulate the research question this thesis addresses. Subsequently, I explain how I approach this question and which methods I use to answer it.

3.1 Research Question

Can the DOPE pose estimator, which I train with synthetic data, bridge the reality gap for the 6-DoF pose estimation task of multiple cars in a traffic monitoring scenario? Current state-of-the-art methods perform well on data with multiple objects from different types[41, 39, 34] where the objects are very close to the camera (e.g.YCB[5], LineMod[13] LineMod Occlusion[3]) in contrast to the bird-eye view of a traffic monitoring scenario. Will the network be able to perform pose estimation on real data with multiple instances of the same class?

3.2 Approach

The problem of pose estimation has been addressed in various ways. Recent approaches [34, 16, 41, 39] show promising results with deep CNNs on this task. The most promising result were obtained by Tremblay et al.[39]. They have shown that they can estimate the pose of multiple different objects in an image in almost real time. In contrast to Tremblay et al.[39], one of the problems of this thesis is to estimate the pose of several objects of the same type, namely cars in traffic. DOPE[39] is based on a deep CNN whose performance depends on the availability of large amounts of annotated data. Due to the many variables present in a 6-DoF pose, the process of obtaining such data in the real world is tedious and error-prone.

As a result the available traffic related pose estimation datasets cover limited perspectives [11, 33, 10, 4, 1, 17, 21, 24]. Also training data is highly correlated with test data (same camera, same object instances, similar lighting conditions) in these datasets[39]. This leads to bad generalization capabilities of networks trained with this data. All available datasets are, in one way or another, not suitable for the problem of

this thesis. Because of that, I create the MultiCarPose pose estimation dataset, which is composed of two subsets:

- domain randomized[36](DR) data and
- photorealistic (PR) data.

The central hypothesis of domain randomization is that if the diversity in a simulator is significant enough, that a model trained with this data will generalize to the real world with no additional training[36, p. 1]. To increase the diversity I will use a multitude of 3D models of different car types and render them with randomized settings in front of randomized background. In general, the rendering of this set is not intended to be photorealistic or physically plausible and follows the recommendations of Tobin et al.[36].

Since previous works[39, 37, 36] recommend a combination of domain randomized and photorealistic data to bridge the reality gap, the second part of my dataset will be in a photorealistic manner. I will generate this data with CARLA[8], the open-source simulator for autonomous driving research. This data is also generated with the goal of high diversity. MultiCarPose includes the ground truth pose annotations for every car in all images, structured similar to the Falling Things dataset[38]. The synthetic data is then utilized for the training and validation of DOPE[39]. In order to evaluate whether the reality gap could be bridged, I will test the model on real images. But I am still challenged by the problem of insufficient data, since synthetic data is not suitable in this case. Therefore, I manually annotate real images for the test set.

The images were taken by two cameras at intersections in China. Apart from projected cuboids I am not capable of generating further ground truth information about the pose. Besides the intrinsics, I have no further information about the images and the environment in which they were taken. Therefore, I cannot determine the position and orientation in the world or in the camera coordinate system.

This fact limits the choice of metrics I can use to evaluate the performance of the network. Originally, I planned to use the average 3D distance of model vertices (ADD) as proposed in[3], the Average Orientation Similarity (AOS)[10] and the Average Precision (AP)[10]. However, for the first two a ground truth pose information is necessary. The latter measures only the overlap of the projected cuboids, which in my opinion is insufficient for a statement about the correctness of a pose.

In order to measure the performance of the network nevertheless, I modify the ADD metric to measure the distance in the 2D image space. Using this ADD-2D metric, I then calculate precision and recall of the network on the test set in regard to different thresholds. Which I will explain in detail in the following chapter.

3.3 Methods

In this section I discuss the methods I use in my approach to answer my research question. Some of these methods have to be applied because DOPE was originally developed for a different domain with different requirements. So the problem of assigning estimations to ground truth simply does not exist if only distinct objects are estimated. Other methods are needed because of the lack of ground truth information, such as the ADD-2D metric.

Kuhn-Munkres Algorithm Since several objects of the same class appear in the image, each pose estimation must be assigned a ground truth. Since errors occur during the pose estimation and therefore an estimation cannot always be assigned to a ground truth, the assignment is not trivial. To be efficient even with many cuboids, I am using the Kuhn-Munkres algorithm also known as the Hungarian algorithm. It is a combinatorial optimization algorithm that solves the assignment problem in polynomial time $O(n^3)$.

For N ground truth cuboids and M cuboid estimations an $N \times M$ matrix is formed. At each position in the matrix the distance between a ground truth cuboid $n \in N$ and an estimation $m \in M$ is entered. The distance between n and m is calculated with the ADD-2D metric. From this matrix the Kuhn-Munkres algorithm generates a list L of tuples (n, m), so that the total distance is minimal for all assignments. Since N and M can have different dimensions, it is possible that $\exists\tilde{n}$ or $\exists\tilde{m}$ so that $\tilde{n}$ or $\tilde{m} \notin L$. These cases are explicitly examined in the implementation and evaluated accordingly.

There is also the possibility that there are several minimal assignments, where the algorithm outputs only one of them. The Kuhn-Munkres algorithm is designed for $N \times N$ matrices only. This is achieved by padding the $N \times M$ matrix.

ADD Metric Originally, I planned to utilize the average distance metric (ADD)[3] for performance measurement. In the following, I will explain how the ADD metric[3] is calculated and why I cannot use it. Afterwards I will introduce the 2D adaptation ADD-2D.

For each detected object DOPE yields a rotation matrix $\tilde{R}$ and a translation vector $\tilde{t}$. Given the ground truth rotation R and translation t, the average distance metric computes the mean of the pairwise distances between the points of the 3D model transformed according to the ground truth pose and the estimated pose [3].

$$ADD = \frac{1}{|P|} \sum_{p \in P} \left\| (Rp + T) - (\tilde{R}p + \tilde{T}) \right\|_2 \tag{1}$$

P denotes the set of 3D model points, $p \in P$ and $|P|$ denotes the number of 3D model points[3].

The pose is considered correct if the average distance is smaller than a threshold. In most cases this is 10% of the diameter of the 3D model.

Estimated Ground Truth The problem in this case is, that I have no ground truth rotation and translation for the images of the test set. A possible approach, similar to DOPE, would be to put the annotated points of the projected cuboid together with the estimated object size into a PnP algorithm to recover poses from 2D-3D correspondences. But this would not result in a real ground truth but in an estimation, since the projected cuboid annotation and also the object dimensions are estimations. The result of the ADD metric, without real ground truth information, depends only on the eight points in the image. It is therefore appropriate to compare these eight points. In addition, during training the network optimizes with regard to the 2D Euclidean distance in 2D image space, therefore it is appropriate to evaluate the performance of the network with respect to this optimization goal.

ADD-2D Metric ADD-2D is a 2D modification of the average distance metric (ADD). In contrast to ADD, no 3D model of the object is required, nor is the object's ground truth pose in the world necessary. Only the eight points of the projected cuboid are sufficient. Unlike the ground truth pose, the projected cuboid can be created manually using annotation tools like CVAT without any additional information about the scene.

I will now explain how the metric is calculated. Let $g \in G$ be the set of points of the ground truth projected cuboid and $g' \in G'$ the set of points of the estimated projected cuboid. Then the metric is calculated as follows.

$$ADD - 2D = \frac{1}{8} \sum_{i=0}^{7} \|g_i - g'_i\|_2 \tag{2}$$

The estimated projected cuboid G' is considered correct if the distance between G and G' in regard to the ADD-2D metric is below dt. Thereby d is the diameter of the 2D bounding box of G and $t \in \{0.05, 0.1, 0.15\}$. I obtain results for all three values of t during evaluation.

$$d = \sqrt{(g_{x_{max}} - g_{x_{min}})^2 + (g_{y_{max}} - g_{y_{min}})^2} \tag{3}$$

Thereby $g_{x_{max}}, g_{x_{min}}, g_{y_{max}}, g_{y_{min}}$ are the smallest and largest x and y value of g in the image space, respectively.

The diameter of the bounding box heavily depends on the pose of the vehicle in regard to the camera. So the diameter is smallest when looking at the car from pedestrian level from the front and largest when looking at the car from above. However, although there may be inaccuracies in these extreme cases, this method is preferable to a fixed threshold that penalizes objects close to the camera and favors objects far away.

Finally, I classify the estimations as true positive (TP), false positive (FP) and false negative (FN). An estimation is a TP, if the Kuhn-Munkres algorithm could assign a ground truth to it, and the distance is smaller than the threshold. An estimation is considered to be FP, if no ground truth could be assigned to the estimation, or the distance is above the threshold. Thereby, I follow the convention of the average precision metric[3]. All ground truths which could not be assigned to an estimation are considered as FN.

Lastly I aggregate these values to calculate precision and recall.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

Precision indicates how many estimations are correct, while recall indicates how many correct estimations were found.

3.4 Summary

The objective in this thesis is to estimate the pose of multiple cars in a traffic monitoring scenario. For this I utilize DOPE which is based on a CNN. Because of the lack of real data for this task, I create a synthetic dataset which should show as much diversity as possible. In order to test performance I cannot use ADD metrics, but have to measure the correctness of the pose in the image. The assignment of estimation and ground truth is performed using the Kuhn-Munkres algorithm.

4 DOPE Pose Estimator

For the task of 6-DoF pose estimation I use the Deep Object Pose Estimator[39]. In this chapter I will discuss some details of the Pose Estimator. DOPE was developed for robotic grasping of household objects and is based on a two-step solution to detect objects and estimate the 6-DoF from a single RGB image. For this a one-shot fully convolutional deep neural network estimates belief maps of 2D keypoints for all objects in the image, which are then fed into a Perspective-n-Point algorithm to estimate the 6-DoF pose.[39, p. 2]

More precise, the CNN detects keypoints using a multistage architecture. The model receives a 640 x 480 x 3 image as input and produces two different outputs, believe maps and vector fields. For each object there are nine believe maps, one for each vertex of the projected cuboid and one for the centroid of the projected cuboid. Additionally there are eight vector fields indicating the direction from each of the eight vertices to the corresponding centroid.[39, p. 2] DOPE utilizes the first ten layers of VGG19[29] pre-

trained on ImageNet[28] as feature extractor, followed by two 3x3 convolutions which reduce feature dimensions from 512 to 256 and from 256 to 128[39, p. 2]. The 128 dimensional features are fed to the first stage of three 3x3x128 layers and one 1x1x512 layer, followed by either 1x1x9 (belief map) or a 1x1x16(vector field) layer[39, p. 2]. The five remaining layers from each branch are identical to the first one, except that they receive 153-dimensional input (128 + 16 vector field + 9 vertices) and consist of five 7x7x128 layers and one 1x1x128 layer before the 1x1x9 (vertices) or 1x1x16 (vector fields)[39, p. 3]. All stages use ReLU activation functions interleaved throughout[39, p. 3]. Extracting the objects from the belief maps, after the network has processed the image, is done by searching for local peaks in the belief maps above a threshold. This is followed by a greedy assignment algorithm that associates vertices to centroids. The object's vertices are fed to a Perspective-n-Point algorithm to retrieve the final pose. For this the PnP receives object dimensions, camera intrinsics and the vertices of the projected cuboid as input. For the training DOPE needs the projected cuboids around the objects including centroid and the object dimensions.

The implementation is based on a ROS node and PyTorch v0.4 and is also available as docker container[3].

5 MultiCarPose Dataset

Pose estimators based on neural networks overcome the pose estimation problem by using large amounts of data with ground truth information. How this data, namely the MultiCarPose dataset, is generated and which information the pose estimator can utilize, is the subject of the following chapter. More specifically, I will first discuss the terms domain randomized and photorealistic data, the connection and differences to the Falling Things dataset and the structure, composition and objectives of the dataset. In the following chapters I describe in detail how I create the respective parts of the dataset and what properties these parts have.

General Dataset Approach MultiCarPose is created for the purpose of training a neural network to perform pose estimation of several cars in a traffic monitoring scenario. The overall goal is to generate a large amount of data in which cars in different environments are represented in highly diverse ways. The basic idea is that a network which uses this data for learning, might be able to generalize easier and be more resistant to overfitting.

The work of Tremblay et al.[39] used a combination of domain randomized data and photorealistic data for the purpose of robotic grasping and achieved promising results. The central hypothesis of domain randomization is that, if the possible diversity in a

[3]https://github.com/NVlabs/Deep_Object_Pose

simulator is significant enough, then models trained with synthetic data from this simulator will generalize to the real world with no additional training. In order to achieve this high diversity, the positions of the objects, the camera and the light are determined randomly for each image when generating domain randomized data. In addition, there is random direction and intensity of light, random color of the textures of the cars, random backgrounds, and much more. In general the rendering of this set is not intended to be photorealistic or physically plausible and follows the recommendations of Tobin et al.[36].

In order to facilitate the transition to real data for the network, not only domain randomized data, but also photorealistic data, is generated with the ambition to reproduce reality The laws of physics apply, both in terms of lighting, textures, and movements. Objects, namely cars, are part of a traffic simulation in which they try to behave like real traffic participants. Different types of vehicles show different behaviour. Some are more likely to cause accidents. In addition, there are even pedestrians and motorcycles and the colour and position of the light source are based on the real world. As with the domain randomized data, I try to create as much diversity as possible.

Structure The structure and annotation format of MultiCarPose is based on the Falling Things (FAT)[38] dataset. FAT is a synthetic dataset for 3D object detection and pose estimation in context of robotic grasping from 2018. The dataset contains 60k annotated images of 21 household objects taken from the YCB[5] dataset. Each image is annotated with 3D poses, per-pixel class segmentation and 2D/3D bounding box coordinates of all objects. Moreover for each RGB image there is also a depth image.

Despite the similarities with FAT, there are some key differences, mainly related to the differences in the selected objects. Instead of 21 household objects, where each object has exactly one 3D model, in MCP there is only one object class (car) which is represented by 18 different 3D models. These different 3D models represent cars with very different looks and dimensions. Among the 3D models are not only cars, but also a truck. The truck acts as kind of advanced distractor and should make sure that the model learns the difference between car and truck explicitly.

In connection with the dataset this leads to another difference compared to FAT and Tremblay et al.[39]. A separate model was trained for each of the household objects. For this purpose for each object 60k images were created in which the object appears alone and another 60k images in which many of the household objects appear mixed together. Since there is only one class in my use case, I decided to produce only images in which at least one, but usually several cars appear simultaneously. Each image is associated with an annotation file that contains the camera pose and a list of all objects in the image. For every object,

- the class,

- the position and orientation in the world,
- the position, orientation and field of view of the camera,
- the eight vertices and the center of the projected cuboid in the image,
- and the vertices and the center of the cuboid in the world

are contained in the annotation file.

Besides the images and the annotation files, there are two files with meta information in the directory. The _object_settings.json contains information about the objects used. In this case these are the dimensions of the car in the world and the fixed_model_transformation, the transformation of the 3D model when creating the data. The _camera_settings.json contains the intrinsics of the camera and the image size.

Training, Validation and Test Set In total more than 113k synthetic images were generated. Of which 50 % are domain randomized images and 50 % photorealistic images. As a common practice in deep learning the synthetic data will then be divided into training and validation set in a ratio 90/10. To assign the files to one of the sets I created a script in which the files are randomly assigned to one of the two sets. The training and validation set consists entirely of synthetic data, while the test set contains only real data. The real data comes from two different intersections in China and was annotated manually. After each training epoch the validation set is used to calculate the loss for data that is unknown to the model. This allows me to make better decisions regarding the hyper parameters and thus avoid over or underfitting the model to the training data.

5.1 Photorealistic Data

I created the photorealistic dataset with the open-source simulator CARLA[8]. CARLA is designed for autonomous driving research and simulates a realistic traffic situation. Physical laws are followed in the simulator, cars drive on the road and can collide with each other. There is a light source simulating the sun, different weather conditions and different types of cars show different behaviour. In addition to motorcycles, pedestrians and cyclists, there are 20 different four- wheeled vehicles in CARLA 0.9.9, of which 17 meet the definition of car in chapter 3.2:

- Audi A2,
- Audi TT,
- Mercedes-Benz Coupe,
- BMW Grand Tourer,
- Audi eTron,
- Nissan Micra,
- Lincoln MKZ2017,
- Dodge Charger,
- Tesla Model 3,
- Toyota Prius,
- Seat Leon,
- Nissan Patrol,

- Mini Cooper,
- Jeep Wrangler Rubicon,
- Mustang Mustang,
- Volkswagen T2,
- Chevrolet Impala and
- Citroen C3.

CARLA is designed as a server-client architecture. The client can control actors, which are managed by the server. CARLA comes with sample files to illustrate the communication between client and server. spawn_npc.py is one of these files and serves as basis for my DatasetCreator class. It utilizes the synchronous mode in which the client controls the progress of the simulation. After setting up some basic features like the traffic manager the DatasetCreator sets attributes for the blueprint of the camera which I place into each map between 40 - 60 times.

Figure 1: Photorealistic Samples

Camera Settings and Positioning As recommended in [39] the image dimensions are 640 x 480. Apart from this the camera makes an image on every simulation tick, postprocessing effects are enabled and all other attributes have their default values. In the first version of the class, one camera per waypoint and traffic light was positioned and aligned in different directions. But in CARLA there is no way to determine if an object is in the field of view of the camera or not. This has led to the fact that even cars behind walls have been detected and annotated. Therefore I had to change the way of positioning the cameras. I extended the BasicSynchronousClient, which is also part of the sample files, so that I can iterate over the positions of traffic lights and waypoints one after the other via shortcuts. To prevent cuboids from appearing behind walls I manually set for each of these positions the height, the orientation and the maximum detection distance. This way I place 40 to 60 cameras on each of the eight maps and save these configurations in a yaml file which the DatasetCreator utilizes. A similar problem occurs when the z-value that determines the height of the camera is low, so that the camera is on pedestrian level. Pictures taken from these low angles, especially near traffic lights, show cars lined up behind each other in such a way that they are heavily occluded. As already mentioned, the degree of occlusion cannot be determined in CARLA. Therefore I avoid camera configurations with low z-values, which make such heavy occlusions possible.

Figure 2: Challenges in CARLA
Car (behind car in the foreground) is not visible but an annotation is generated (left image).
Car is visible but not an actor, so no annotation is generated (right image).

Another problem I face during the generation of the dataset with CARLA is, that in some maps, static cars are located in parking lots that are no active actors, but part of the map, like a house. When selecting the positions of the cameras, I had to avoid having these static cars in the field of view of the camera. Because they look just like other cars, but are no active actors and are therefore not labeled as cars during annotation. Missing labels on cars would impede the training and lead to poorer results and therefore should be avoided.

Actors After settings up the cameras the DatasetCreator spawns all actors. The set of actors is made up of cyclists and motorcyclists, pedestrians and cars. From the set of all vehicle blueprints those are removed which are untypical vehicle types. Among them are Tesla's Cypertruck because of its untypical look, the BMW Isetta because of its size and the police car because of the flashing lights and siren. While trucks, bicycles and motorcycles remain, they will not be annotated. Each of the eight maps has a fixed number of spawn points. In order to allow traffic flow and thus record cars at different places on the map, only on half of the available spawn points vehicles are spawned. Additionally there are 80 pedestrians spawned.

Data Generation After the cameras are positioned and all actors on the map are spawned and set in motion, data generation can begin. A time step in CARLA is called a tick. At each tick, all cameras on the server take an image and send it to the client. Since all images come at the same time, this leads to concurrency, which I could solve by giving each camera its own image queue. After the images were fetched from the queue they had to be re-associated with the camera. Since both, the camera object and the image object contain their pose, I created a camera dictionary for this purpose that assigns a camera object to each pose. So I could assign a camera to a picture and save the picture in the camera object and process it.

The processing takes place in the cuboid creator which receives a list of vehicles to detect, a list of cameras and the maximum distance value. Practically, there is already an implemented functionality in CARLA to create cuboids for vehicles. To get all necessary information for the pose estimation it had to be adjusted. Of all the vehicles in the simulation, only those which meet the definition of a car should be annotated. Therefore, I remove all motorcycles, bicycles and the Carla Cola truck from the list of vehicles to be annotated.

For all remaining vehicles it is checked whether they are within the detection range, i.e. not too far away from the camera or even behind the camera. As soon as a point of the cuboid is in the image, an annotation is created for this vehicle. With the information described in chapter 5 the following had to be considered. The position in CARLA is given in meters, but in FAT it is given in centimeters. Therefore, all position information must be converted into centimeters. The order of the points of a cuboid in CARLA does not correspond to the order in FAT and must therefore be rearranged. In CARLA, angles are always output as Euler angles. They must therefore be converted into quaternions, as is usual in FAT. The annotations are collected for all vehicles in an image and then saved in JSON format along with the image. The creation of about 57k photorealistic images with ground truth information took about 22 hours with my implementation.

Diversity In order to achieve the goals mentioned in chapter 5 several measures were taken. First, I use all eight default maps available, in which different scenarios are displayed. These include large cities, rural and tropical areas. Due to a bug in the traffic_manager no new traffic could be generated after a map change. I solved this problem by writing a bash script that starts CARLA, changes the map, executes create_data.py, which contains the DatasetCreator, and after successful data generation, restarts CARLA for the next map.

Secondly, the weather is changed after each simulation tick. Of the fifteen presets, only six are suitable. In the others, the vehicles are not visible with the naked eye, because either the low position of the sun leads to large dark shadows on the road (ClearSunset), or the heavy rain and overcast skies generally make the scenario too dark (HardRainNoon).

Furthermore I tried to place all available vehicles in the simulation. The vehicles were also assigned a random color. Despite the problems described in chapter 5.1, I tried to choose as many different perspectives of the cameras as possible. After everything is set up it takes about 24 hours to create 55k images with annotation files.

5.2 Domain Randomization

I use the NVIDIA Deep learning Dataset Synthesizer[4] (NDDS)[35] to generate the domain randomized data. NDDS is a plugin for Unreal Engine 4 from NVIDIA which allows users to create highly randomized synthetic images with metadata in Falling Things Dataset[38] format. It supports images, segmentation, depth, object pose, bounding box, and more[5].

Objects Coincidentally, CARLA is also based on the Unreal Engine 4 and it is possible to open CARLA in the Unreal Editor. So it was easy for me to select the 3D models of the vehicles used in CARLA and migrate them to NDDS. Except for some missing textures this worked without problems. An object in NDDS to be annotated must be an AnnotatedActor. The class DR_AnnotatedActor_BP is available for this purpose. It already contains all the basic properties such as RandomMovement, RandomRotation and an AnnotationTag. Only the mesh had to be exchanged.

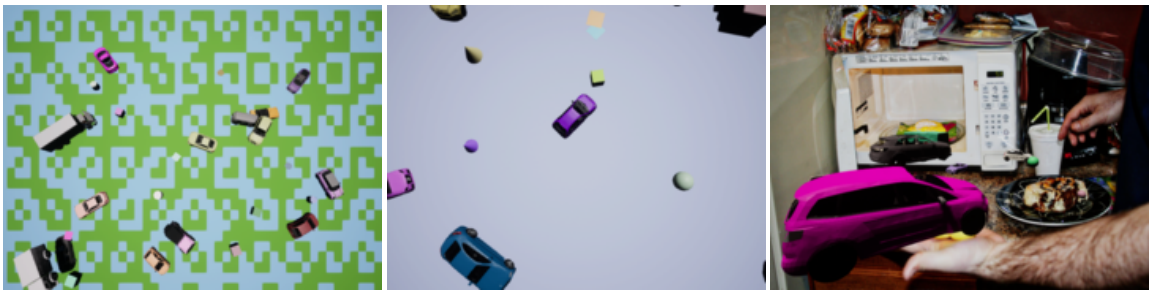


Figure 3: Domain Randomization Samples

Level When creating a level I followed the documentation of NDDS. The level consists of a sphere with a diameter of about 40 meters. Within this sphere the whole process takes place. First I created two volumes, one which contains distractor objects while the other contains the vehicles. Fourteen different cars are spawned in the volume and are randomly teleported to a different location within the volume with each tick. The distractor objects all start at a common location and fly randomly through the volume. Above the volumes is the camera SceneCapturerSimple, which takes pictures with 640 x 480 pixels and has a field of view of 90 degrees. As extraction features I select only the object data and true color images. Depth, Instance Segmentation and Class Segmentation are omitted, since they are not necessary for the pose estimation task.

Diversity In order to achieve a high variance some measures were taken. So all vehicles get a random paint color after each tick. Vehicles are not only teleported

[4]https://github .com/NVIDIA/Dataset_Synthesizer
[5]I used NDDS in version 1.2.2 based on Unreal Editor 4.22.3.

randomly but are also rotated in a random yaw angle, so they turn in a random direction standing on an imaginary floor. As a result, the camera never shoots vehicles from below. This is also because the camera is positioned above the volumes and focuses a focal point in the middle of the volume. Beyond that, the camera randomly orbits this point and changes the distance and the angle randomly. The background also changes randomly. It is either one of about 11k images from ImageNet dataset as [39] recommended, a checker pattern of two random colors or a single random color. In addition, the light in each image also changes. The light intensity oscillates between 1.5 and 5 units and the position and rotation of the light source also changes constantly.

Problems By default an AnnotatedActors has a tag. The name of the tag is derived from the name of the 3D model mesh. This leads to many different values in the annotation files, although there should be just *car* for all objects. I solved this problem with the help of a script afterwards, where I changed the class to "car" in all annotation files.

Data Generation After the level, all actors, the distractor objects and the random light were set up, I could start the process of data generation. With a very high frequency, randomized scenes are generated and image and annotation data is generated. The generation of 57k domain randomized images took about 60 minutes.

5.3 Test Set

For the purpose of measuring the performance of the network trained with synthetic data, I created a test dataset of 144 real images with 336 car annotations. The images are extracted from heavily compressed video recordings of two intersections in China. The cameras are mounted on high buildings in Shanghai and Huainan. The images are rectified afterwards. Therefore, I used a script based on OpenCV that takes an image and the associated camera intrinsics as input and returns the rectified image. The network takes images with size of 640 x 480 px as input.

Shanghai Huainan

Figure 4: Source of Test Data

Due to the high resolution[6] of the recordings I had to decide whether I would shrink the image or use only parts of these images by cropping them.

Because of the high position of the cameras the cars are very small relative to the whole image which can be seen in Figure 4. Shrinking the image would lead to even fewer pixels per car, which also means fewer features to recognize for the network. On account of the big image size I initially choose four areas on each intersection to crop test images from.

Figure 5: Sample of Test Data
Shanghai (top), Huainan (bottom).

Annotation These cropped images are manually labeled with the help of OpenCVs Computer Vision Annotation Tool (CVAT)[7]. CVAT allows cuboid annotation without further data (e.g. multi sensor systems[8] or geopgraphic modelling[9]).

For reasons I explained in chapter 3.3, the annotation data in the test dataset will differ from that in the training and validation set. This means that the annotation data only contains all objects and their "projected_cuboid", i.e. the vertices of the projected cuboid. Apart from the projected cuboids, none of the metadata described in chapter 5 can be captured with CVAT. This includes the pose of the camera and ground truth pose of the vehicle.

Even if it is only about the projected cuboid, the task of image annotation for the pose estimation task is tedious and error-prone. Although CVAT is a very sophisticated and powerful tool, the creation of cuboids is somewhat limited. Not all edges of the

[6] Camera in Huainan 4096 x 2160px, in Shanghai 2560 x 1440px
[7] The sourcecode is found https://github.com/opencv/cvat
[8] 3D Bounding Box Annotation Tool (3D-BAT) needs data from a multi sensor system mounted on a roof of a car
[9] Matzen and Snavely[20] created the nyc3dcars-labeler which uses a geographic model of the location the image was taken, to annotate cuboids on the ground plane.

cuboid are freely movable, because one of the edges must always be vertical. As a result vehicles in certain poses could not be correctly annotated at all. Which leads to a further reduction in the originally planned extent of test data, as many images had to be discarded.

Metadata Conversion CVAT offers the possibility to export the metadata of annotated images in different formats. Unfortunately the FAT data format was not among them, so I had to develop a CVAT to FAT converter. The main task of this tool was to put the corner points of the cuboid output by CVAT in the correct order and save the annotations in a way that is compatible to FAT[38]. With FAT, the order of the points depends on the pose of the object. For example, the point 0, from the driver's perspective, is the upper right point above the hood. In order to be able to rearrange the order of the points correctly, I had to annotate some additional information. CVAT marks the front surface of the cube from the annotator's perspective. Although it is possible to select a different surface, this had no effect on the order of the points. So as additional information, I have indicated on which side this marked surface is located from the driver's point of view. So it is possible to identify all points clearly afterwards.

5.4 Summary

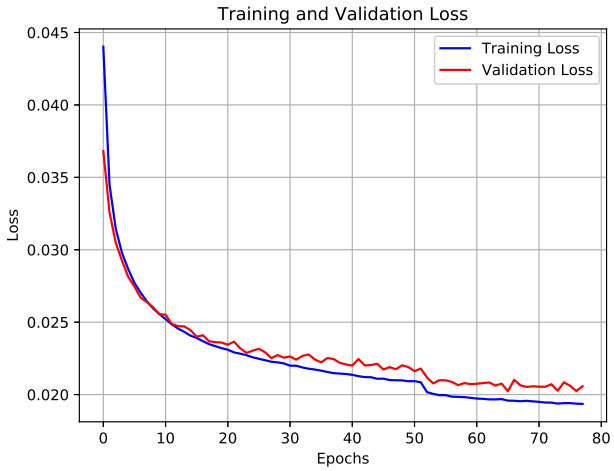
I generate half of the data for training and validation from photorealistic and half from domain randomized images. The main focus is always to include as much diversity as possible. With the help of CARLA and NDDS I was able to generate more than 113k images and ground truth information in less than a day. The data for the test comes from two crossings in China and is manually annotated and does not contain any real pose information. The complete dataset is available online[10].

6 Training

In this section I focus on the progress and the chosen parameters for the training. Approximately 100k images were used for the training and 10k for the validation. These are composed of domain randomized and photorealistic images. For the data augmentation, I use the default settings of the DOPE implementation. These are in detail Gaussian noise, random contrast and random brightness. As loss function DOPE uses the L_2 Loss for belief maps and vector fields. The loss function is the target function which is to be optimized during training. The pre-trained weights for VGG-19 come from the torchvision open models. The network was trained for 77 epochs total. The graph shows that the training and validation loss is continuously decreasing and the overall training is converging. I used a learning rate of 0.0001 for

[10] http://bit.ly/MultiCarPose

60 epochs. The learning rate can be understood as the step width of the optimization. Since the training and validation loss started to diverge from epoch 52 I reduced the learning rate to 0.00005 for the last 25 epochs. As a consequence, the loss decreases more significantly.



The network is trained with the help of eight GeForce GTX 1080 Ti, each with 11GB RAM. This allowed me to reach a maximum batch size of 96, which is constant for the whole training.

7 Evaluation

This chapter deals with the evaluation of the performance of the network in regard to the test set. I describe my expectations, results of the measurements and describe observations during the analysis of the output images.

7.1 Test Setup

DOPE is implemented as a ROS node. To measure the performance of models I implemented another node Gauge, which reads the images of the test set one by one and sends them together with the intrinsic of the camera via ROS message to the DOPE node. It utilizes the previously trained net and performs the pose estimation. The results are then sent in single ROS messages to the Gauge for evaluation. I also adapted the implementation of DOPE to output not only the pose but also the estimated projected cuboids of the objects. These, I compare with the ground truth by utilizing the Kuhn-Munkres algorithm for the assignment of estimations to ground truth projected cuboids to finally collect the results for the evaluation. The test dataset is divided by origin of the images. The two groups, one from Huainan and one from Shanghai, differ in several aspects.

Assumptions Due to the size and quality of the dataset and the promising results of Tremblay et al.[39] I expect that the model will be able to recognize the pose of multiple cars in the picture. I assume, however, that it has difficulties with multiple close pose estimations. Also cars located far away, which have less surface area and therefore less features in the image, might not be recognized by the model. This is probably the case with the pictures from Shanghai, where the cars very small in certain areas. I also assume that I can utilize the ADD-2D metric to evaluate the performance of the model in terms of pose estimation without relying on a real ground truth poses.

For the evaluation I calculate precision and recall for three different thresholds. As already described in Section 3.3, these are 5%, 10% and 15% of ground truths diameter. If the distance between ground truth and estimation according to ADD-2D metric is smaller than the threshold, this estimation counts as true positive, otherwise as false negative or false positive (see Section 3.3). For each of the three thresholds I calculate precision and recall, see table 1.

In order to illustrate how projected cuboids appear that meet different thresholds, I have created an overview in Figure 6.

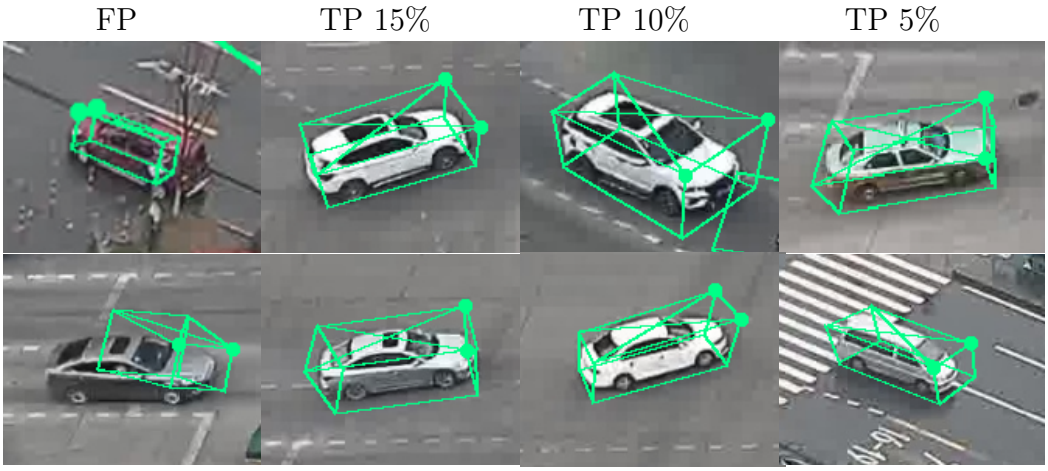


Figure 6: Projected Cuboids Thresholds
Examples for projected cuboids below certain thresholds.

Observations Before I come to the evaluation of the numbers, I want to describe some observations I made while looking at the results on the pictures. The first thing I notice is that it seems to make a difference which color a car has. So white or bright colored cars are very often recognized. Black or very dark cars, on the other hand, are almost never recognized by the model. The effect is more obvious in the pictures from Shanghai than in the pictures from Huainan.

Secondly, the model has problems with symmetry in some type of cars. For cars with a hatchback there is some kind of an axis of symmetry on the B-pillar. This leads to points of the projected cuboid that should be above the hood being placed above the trunk, which is counted as FP at all three thresholds.

	Huainan		Shanghai		Total	
Threshold	Precision	Recall	Precision	Recall	Precision	Recall
15%	0.80	0.71	0.86	0.37	0.83	0.51
10%	0.54	0.63	0.73	0.33	0.62	0.44
5%	0.17	0.34	0.31	0.17	0.22	0.22

Table 1: Results on MultiCarPose Test Set

In table 1 the large differences between the thresholds are apparent. The total precision varies between 0.22 and 0.83, in Huainan even between 0.17 and 0.80. Thereof, I can conclude that the estimations are not always very tight around the car. This was also noticeable when evaluating the pictures. The range between the recall values is on the other hand not that wide. Overall, the recall is low. In Huainan it is much higher than in Shanghai.

7.2 Discussion

Two points regarding the results are surprising. One is the wide range between the thresholds in terms of precision and the other is the low recall.

Low Recall When looking at the pictures, the bad recall was already noticeable. There are many cars that are not recognized. Especially the recall in Shanghai is very bad. The images in the test set have a high level of noise. So high that it even made annotating to be difficult. In addition, the images are so highly compressed that many artifacts are formed and the facets of dark cars are barely visible. Also, the road in Shanghai is particularly dark, which means even less contrast for dark vehicles. The greater distance of the camera in Shanghai certainly contributes to the low recall too. Larger distance means fewer features for cars in the image for the pose estimation. Already during the evaluation of the validation and test set, it was noticeable that the model rarely recognizes small vehicles at long distances. Looking at the images from Huainan, a brighter road and vehicles at a shorter distance from the camera are noticeable. Another reason for the bad performance with dark vehicles could be the training data. In it, the boundaries of the vehicles are always clearly separated from the background. These boundaries between object and background are features that feature extractors learn in neural networks. Unfortunately, these features are only found in the test set for bright colored, but less so for dark cars.

Wide Precision Range Secondly, the wide range in terms of precision is surprising. Depending on the threshold applied, the precision is between 0.22 and 0.83. Considering the images, the truth must be somewhere in the middle, probably rather near 0.83. Of course there are some outliers and wrong estimations, but the majority of the estimations I would consider as correct. The wide range also seems to be a metric problem

rather than a model problem. As can be seen in Figure 6, the estimation must be very close to the car to be considered as TP at 5%. The projected cuboids in the test set are annotated rather closely around the car. As a consequence, many estimates fail the 5% threshold and pass the 15% threshold. This may explain the poor results for the 5% metric and the wide range too. So the problem is in the ADD-2D metric. It uses the projected cuboids as a ground truth and therefore depends on them being to be very similar with the estimation. But in the training set the projected cuboids are rather wide around the object, whereas in the test set they are rather tightly. Therefore it is important to look at all three thresholds together. In this way it is possible to infer how closely fitted the projected cuboids are. Despite this disadvantage of ADD-2D, it is still possible to make a statement about the performance of the model. It is especially important not to set the threshold too high, otherwise outliers can be evaluated as TP.

ADD-2D Significance I would like to address the question to what extent the tests are significant and whether I can make a statement about the performance of the Pose Estimator. As I have described in chapter 3.3, I have to abandon the ADD metric for this thesis.

As described in chapter 7.2 the ADD-2D metric is very sensitive to the size of the projected cuboid. The metric can be used to measure the performance of DOPE and other pose estimators that output the projected cuboids. However, it is important to ensure that the projected cuboids are similar in the training data and in the test data. Otherwise problems will occur as described in chapter 7.2. Considering multiple thresholds can help to detect those and to make a more precise statement.

Nevertheless, I can make statements about the quality of the poses and therefore about the performance of the model with the ADD-2D metric. Since the eight points of the projected cuboid are representatives of the pose. During training the network also optimizes with regard to distance in image space therefore it is appropriate to evaluate the performance of the network with respect to this optimization goal.

Hypothesis Evaluation Which of my assumptions have proven to be valid? My assumption that there are big issues regarding the distance is only partially correct. Distance plays a role especially with dark cars. In the results some light cars have been detected at a large distance, but not a single dark one. So the color of the car and the contrast to the environment plays a big role. Furthermore, contrary to my expectations, the results in Huainan are not much better than in Shanghai. However, an assumption that has been fulfilled and is therefore one of the most important insights, is the core assumption of this work. The model, which has learned the pose estimation from synthetic images only, is able to estimate the pose of multiple cars in a traffic monitoring scenario. Even if the 5% threshold is used, the model is able to estimate the pose of several cars in one image.

Applicability To what extent can the model be used in the real world? That depends strongly on the area of application. In general, the technical requirements for hardware and software in traffic are very high. The model would probably not meet these requirements in its current state. For example, when used to detect dangerous situations on intersections, it would not be suitable for two reasons. The recall is too low for a reliable use. Too many vehicles would not be detected. The precision is also too low, which would lead to false alarms. However, the model is suitable for use in research, as a supplement to existing systems or in cooperation with other sensors or technologies. Furthermore, there are still many potential ways to improve the performance. These include modifying the training dataset to include more dark cars and scenes with low contrast. Additionally, noise could be increased. On the other hand, you could also try to reduce the compression of the test data and thus reduce the noise.

8 Conclusion

In this thesis I have shown that the DOPE Pose Estimator can utilize synthetic data to estimate the pose of several cars in a traffic monitoring scenario. Since there is no dataset for this scenario, I first generated a dataset of synthetic images containing cars in different poses and environments. To increase performance and robustness, this dataset consists of both, realistic and highly randomized images. I then used this dataset for training the DOPE pose estimator. For the assignment of an estimation to a ground truth during testing, I utilized the Kuhn-Munkres algorithm. Finally, I was able to apply the model to real data and show that it can bridge the reality-gap and estimate the pose of multiple objects of the same class in one image. Due to the lack of ground truth information I developed the ADD-2D metric, that allows to evaluate the performance even without the ground truth pose. One of the insights, apart from answering the research question was, that this metric is very sensitive to the size of the projected cuboid. Furthermore it is still open to what extent the proposed modifications in Section 7.2 can improve the performance. One of the special features of the DOPE pose estimator is, besides its accuracy, the almost real time performance. A question that might be approached in the future is how the detection of multiple objects of the same class influences this real time performance. My research has contributed a metric to measure the performance of pose estimators in image space without the ground truth pose. Furthermore, I was able to use the Kuhn-Munkres algorithm for the assignment of estimations and ground truth. Previously the DOPE pose estimator was used only for robotic grasping of different household objects. This thesis showed that it can be used in other domains where several objects of the same class occur.

Bibliography

[1] K. Behrendt. Boxy vehicle detection in large images. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 840–846, 2019.

[2] E. Bochinski, V. Eiselein, and T. Sikora. Training a convolutional neural network for multi-class object detection using solely virtual world data. *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 278–285, 2016.

[3] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*. Springer, 09 2014.

[4] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[5] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. pages 510–517, 07 2015.

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[7] T.-T. Do, M. Cai, T. Pham, and I. D. Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image. *ArXiv*, abs/1802.10367, 2018.

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. Visual object classes challenge 2012 dataset (voc2012). 2012.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[11] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth. A2D2: AEV Autonomous Driving Dataset. `http://www.a2d2.audi`, 2019.

[12] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[13] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. volume 7724, 10 2012.

[14] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On pre-trained image features and synthetic images for deep learning. In L. Leal-Taixé and S. Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 682–697, Cham, 2019. Springer International Publishing.

[15] J. Josifovski, M. Kerzel, C. Pregizer, L. Posniak, and S. Wermter. Object detection and pose estimation based on convolutional neural networks trained with synthetic data. pages 6269–6276, 10 2018.

[16] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. pages 1530–1538, 10 2017.

[17] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 av dataset 2019. urlhttps://level5.lyft.com/dataset/, 2019.

[18] S. Khan. *Towards Synthetic Dataset Generation for Semantic Segmentation Networks*. PhD thesis, 09 2019.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. Dec. 2015.

[20] K. Matzen and N. Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *Proc. Int. Conf. on Computer Vision*, 2013.

[21] M. Meyer. Automotive radar dataset for deep learning based 3d object detection. 01 2019.

[22] S. Peng, Y. Liu, Q.-X. Huang, H. Bao, and X. Zhou. Pvnet: Pixel-wise voting network for 6dof pose estimation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4556–4565, 2018.

[23] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1278–1286, 2014.

[24] M. Pitropov, D. M. García, J. Rebello, M. K. Smart, C. Wang, K. Czarnecki, and S. L. Waslander. Canadian adverse driving conditions dataset. *ArXiv*, abs/2001.10117, 2020.

[25] M. Rad and V. Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3848–3856, 2017.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. June 2015.

[27] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[30] C. Song, J. Song, and Q. Huang. Hybridpose: 6d object pose estimation under hybrid representations, 2020.

[31] H. Su, C. Ruizhongtai Qi, Y. Li, and L. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. 05 2015.

[32] B. Sun and K. Saenko. From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[33] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019.

[34] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. Nov. 2017.

[35] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield. NDDS: NVIDIA deep learning dataset synthesizer, 2018. `https://github.com/NVIDIA/Dataset_Synthesizer` .

[36] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. pages 23–30, 09 2017.

[37] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. 04 2018.

[38] J. Tremblay, T. To, and S. Birchfield. Falling things: A synthetic dataset for 3d object detection and pose estimation. 2018.

[39] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects, 09 2018.

[40] D. Vázquez, A. M. López, J. G. Marín, D. Ponsa, and D. Gerónimo. Virtual and real world adaptation for pedestrian detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:797–809, 2013.

[41] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 06 2018.

[42] S. Zakharov, I. S. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1941–1950, 2019.

[43] S. Zhang, C. Wang, Z. He, Q. Li, X. Lin, X. Li, J. Zhang, C. Yang, and J. Li. Vehicle global 6-dof pose estimation under traffic surveillance camera. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159:114–128, 01 2020.

Appendix

My implementation can be found on GitHub[11]. Finally I show randomly selected images from Huainan and Shanghai that the model has produced.

[11] github.com/belorenz/6dof-thesis-code

Figure 7: Results on Test Set 1/2
Random sample from Huainan test set.

Figure 8: Results on Test Set 2/2
Random sample from Shanghai test set.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, 13. September 2020: